Besturingssystemen theorie

Academiejaar 2018 – 2019

HO
GENT

Inhoudsopgave

HO
GENT

Hoofdstuk 1: Inleiding

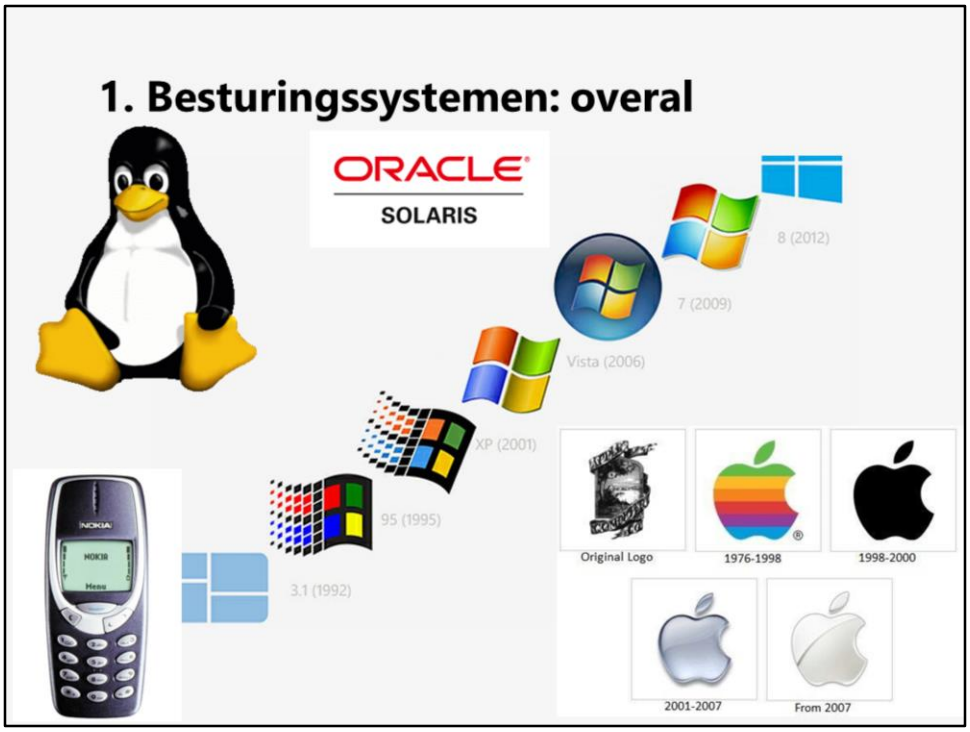
1. Wat is een besturingssysteem?
2. Een kort historisch overzicht
3. Soorten besturingssystemen
4. Concepten van besturingssystemen

HO GENT

In hoofdzaak is een besturingssysteem een programma dat mensen de mogelijkheid geeft gebruik te maken van de hardware van een computer (CPU's, geheugen en secundaire opslagapparatuur).

Gebruikers geven geen instructies aan de computer, maar in plaats daarvan aan het besturingssysteem. Het besturingssysteem geeft de hardware de opdracht de gewenste taken uit te voeren.

De Engelse term voor besturingssysteem is "Operating System", de afkorting OS wordt vaak gebruikt.



Traditioneel denken vele gebruikers dat Windows het enige besturingssysteem is, of dat er slechts één alternatief is genaamd OS-X. Er zijn echter veel meer besturingssystemen.
Onder IT'ers is het Linux besturingssysteem wijd verspreid, een OS dat we verder in deze cursus zullen behandelen. De parallellen met één van de oudste besturingssystemen, Unix, zijn groot. O.a. het bedrijf Oracle verkoopt nog steeds Unix.

Vergeet niet dat er ook heel wat besturingssystemen bestaan hebben, die nu niet meer bijgewerkt worden. Rekenmachines, klassieke GSM toestellen, … hebben ook allemaal hun eigen besturingssysteem, ontworpen voor de specifieke hardware van die toestellen.

Veel mensen willen van systeembronnen (resources) gebruik maken. Als er geen besturingssysteem zou bestaan, zouden er conflicten optreden en zou er een enorme verwarring ontstaan. Een besturingssysteem moet zoveel mogelijk gebruikers gelukkig maken.

Enkele taken van een besturingssysteem:
→ programma's de mogelijkheid geven informatie op te slaan en terug te halen;
→ programma's afschermen van specifieke hardwarezaken;
→ de gegevensstroom door de componenten van de computer regelen;
→ programma's in staat stellen te werken zonder door andere programma's te worden onderbroken;
→ onafhankelijke programma's de gelegenheid geven tijdelijk samen te werken en informatie gemeenschappelijk te gebruiken;
→ reageren op fouten of aanvragen van de gebruiker;
→ een tijdsplanning maken voor programma's die resources willen gebruiken.

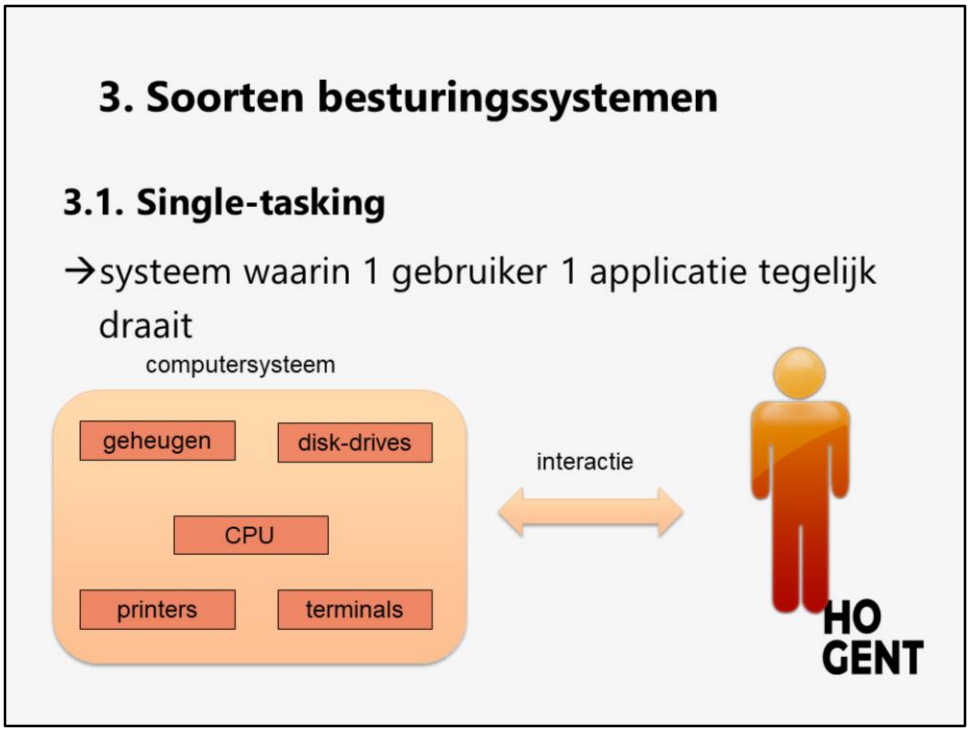
Over het algemeen kunnen we zeggen dat een besturingssysteem zoveel mogelijk gebruikers gelukkig moet maken. Natuurlijk is het geluksgevoel van de gebruiker afhankelijk van zijn verwachtingen.

Een besturingssysteem is een dynamisch iets. Vele geavanceerde systemen van tegenwoordig hebben nauwelijks punten van overeenkomst met de eerste ontwerpen.
De eerste elektronische computer had geen besturingssysteem. Hij moest met de hand bediend worden.

In de **jaren 50** werden er eenvoudige besturingssystemen ontwikkeld waarmee programma's na elkaar in de computer konden worden ingevoerd en opgeslagen. Wanneer het ene programma klaar was, laadde en startte het besturingssysteem het volgende programma. Alle resources van de computer konden slechts door één programma gebruikt worden.
In de **jaren 60** konden verscheidene programma's tegelijkertijd in het geheugen worden opgeslagen. De programma's gebruikten de resources van de computer dan gemeenschappelijk. De programma's werden nu niet na elkaar, maar om beurten uitgevoerd. Elk programma liep een tijdje en dan schakelde het besturingssysteem naar een ander programma over. Hierdoor kwamen de korte programma's eerder aan de beurt, en omdat de resources gemeenschappelijk werden gebruikt, waren ze sneller afgelopen. Een gebruiker kon vanaf een terminal inloggen en vrijwel direct toegang tot de resources krijgen.

In het **midden van de jaren 60** kon één besturingssysteem voor verschillende computers van hetzelfde type worden gebruikt, zodat een upgrade eenvoudiger werd.
In het **begin van de jaren 70** kunnen besturingssystemen computers met meer dan één processor aan.
In het **begin van de jaren 80** zijn besturingssystemen geschikt voor gemeenschappelijk gebruik van informatie door verschillende computersystemen.
De **jaren 90** worden gekenmerkt door begrippen als distributed computing, parallelle verwerking, GUI, ...

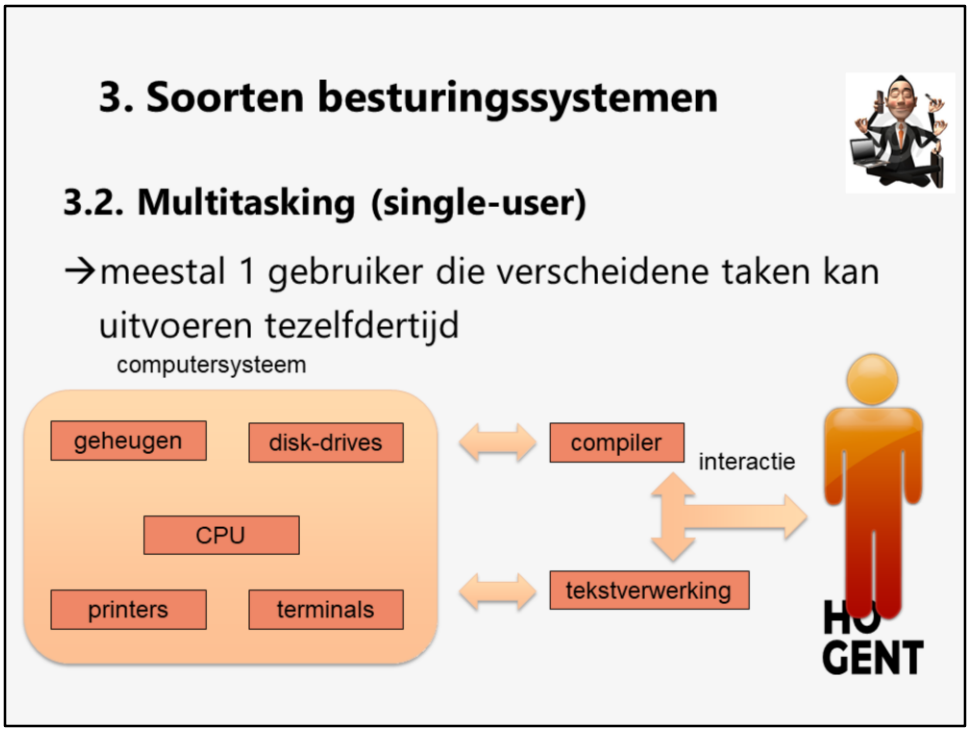
Besturingssystemen zorgen ervoor dat de hardware van de computer bruikbaar wordt.
Ook al is de gebruikersinterface (user interface) van uitzonderlijk belang, er is nog meer. Een besturingssysteem bepaalt waartoe de gebruiker in staat is en hoe efficiënt hij of zij dit kan. Het bepaalt welke hardware op een computer kan worden aangesloten. Verder bepaalt het welke programma's de computer accepteert.
Niet alleen professionele informatici, maar ook mensen die maar af en toe achter een computer zitten, weten waarschijnlijk meer over een besturingssysteem dan over de computer zelf.



Omdat er veel verschillende soorten computers bestaan, zijn er ook veel verschillende soorten besturingssystemen. Sommige eenvoudige besturingssystemen stellen alle resources in dienst van één applicatie tegelijk. Andere geven de gebruiker de mogelijkheid verscheidene applicaties gelijktijdig te laten uitvoeren. Nog ingewikkelder systemen kunnen tegelijk de behoeften van vele gebruikers aan.

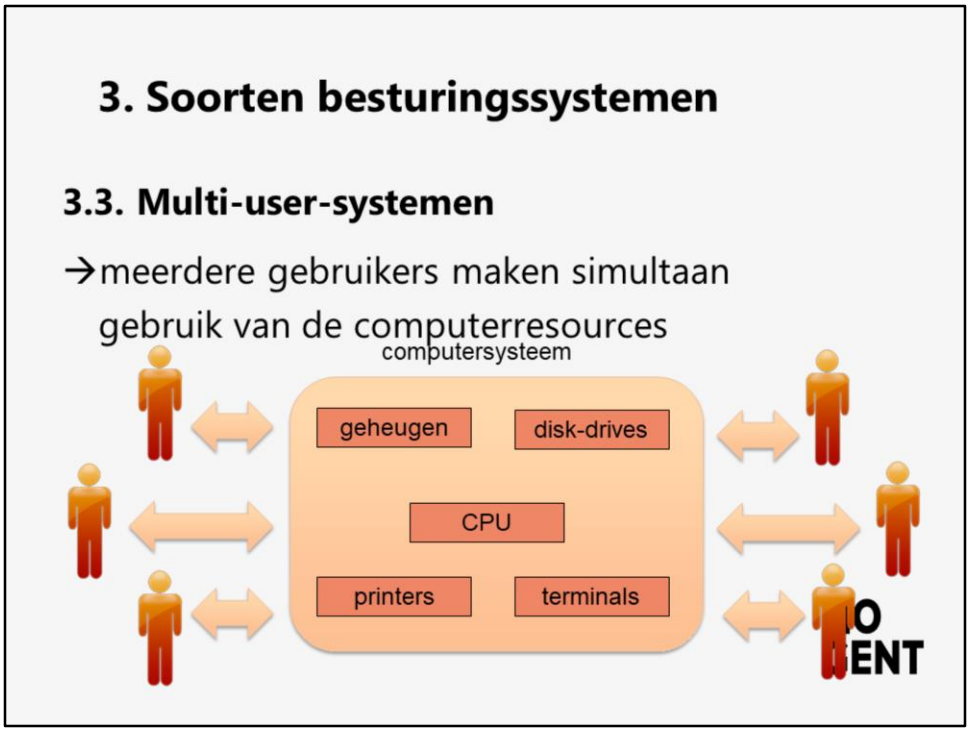
Een besturingssysteem moet echter op meer dingen letten dan alleen het aantal programma's of gebruikers. In sommige gevallen is het voldoende om programma's na elkaar te verwerken. Er is dan geen noodzaak aanwezig om snel te reageren. Aan de andere kant proberen sommige besturingssystemen er voor te zorgen dat snel aan de verlangens van de gebruiker wordt voldaan. In weer andere gevallen moet een systeem binnen een bepaald tijdsbestek reageren om rampzalige toestanden te voorkomen. Alles hangt af van de behoeften van de gebruikers, van wat ze verwachten en van de applicaties die ze draaien.

Single-tasking → Een systeem waarin één gebruiker één applicatie tegelijk draait, heet een single-tasking systeem. Onder dit systeem kan slechts één programma (task) tegelijk actief zijn.



Veel van de multitasking systemen staan nog steeds maar één gebruiker toe, maar hij of zij kan verscheidene bezigheden op hetzelfde moment laten afwikkelen.

Aangezien een gebruiker verscheidene werkzaamheden gelijktijdig kan laten verrichten, **worden bepaalde functies van het besturingssysteem, bijvoorbeeld geheugenbeheer, ingewikkelder.**



Multiuser-systemen, ook wel **multiprogrammering-systemen** genoemd, moeten niet alleen alle gebruikers bijhouden, er moet ook voorkomen worden dat deze elkaar hinderen of in het werk van de ander rondneuzen.

3. Soorten besturingssystemen

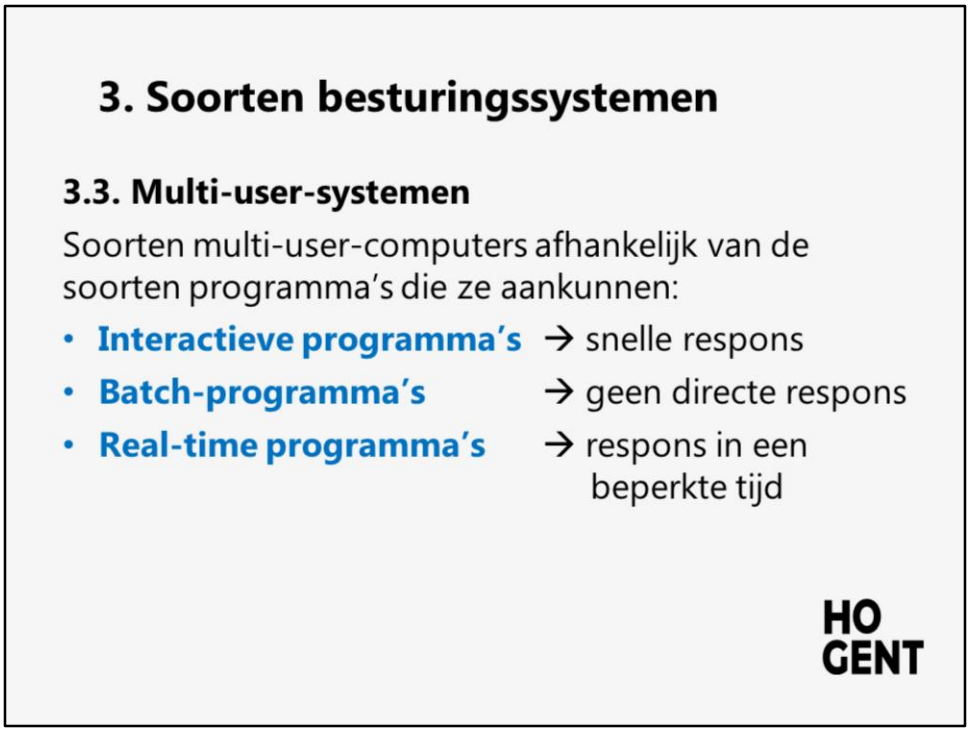
3.3. Multi-user-systemen

→ hierbij is **scheduling** een belangrijk concept

verwijst naar de manier waarop processen prioriteiten worden gegeven in een prioriteitenwachtrij (zie later)

HO GENT

In een multiuser-systeem wordt **scheduling** belangrijker. Bij een single-user computer hoeft het besturingssysteem slechts de behoeften van één gebruiker te vervullen. In een multi-user-computer gaat het om de behoeften van velen. Dit kan moeilijk of zelfs onmogelijk zijn. Aangezien veel programma's de resources van de computer gemeenschappelijk moeten gebruiken, moet het besturingssysteem beslissen wie wat krijgt en wanneer. Vaak zal het aan elk programma een prioriteit toekennen. Maar hoe beslist het besturingssysteem welke prioriteit een bepaald programma krijgt? Elke gebruiker denkt dat zijn of haar programma de hoogste prioriteit zou moeten hebben en de programma's van alle anderen een lagere. Hoe lost het besturingssysteem dit probleem op? Hoe stelt het vast welke programma's werkelijk een hoge prioriteit verdienen?

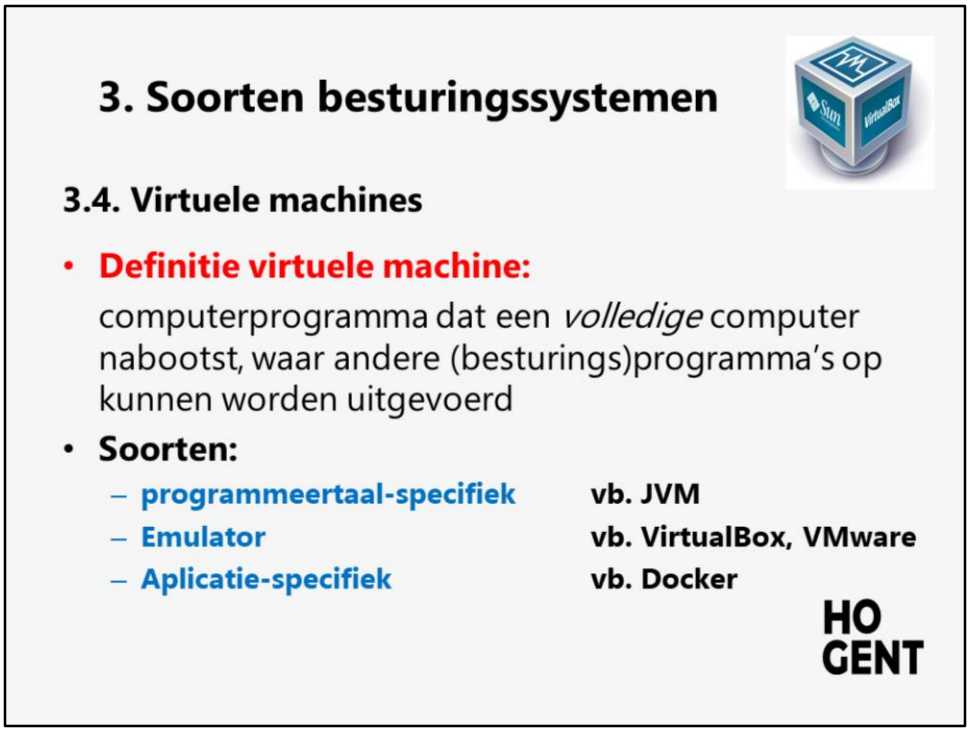


Er bestaan verscheidene soorten multiuser-computers, afhankelijk van de soorten programma's die ze aankunnen:

Interactieve programma's: Een interactief programma is een programma dat een gebruiker vanaf de terminal activeert. Over het algemeen voert de gebruiker een korte opdracht in. Het besturingssysteem vertaalt deze opdracht en onderneemt actie. Het zet vervolgens een prompt-teken op het scherm en geeft daarmee aan dat de gebruiker een volgende opdracht kan invoeren. De gebruiker voert weer een opdracht in en het proces gaat door. De gebruiker werkt met het besturingssysteem op een conversatie-achtige manier, interactieve mode genoemd. Interactieve gebruikers verwachten een snelle respons. Daarom moet het besturingssysteem interactieve gebruikers voorrang geven.

Batch-programma's: Een gebruiker kan opdrachten in een file opslaan en deze aan de batch queue (wachtrij voor batch-programma's) van het besturingssysteem aanbieden. Uiteindelijk zal het besturingssysteem de opdrachten uitvoeren. Batch-gebruikers verschillen van interactieve gebruikers omdat zij geen directe respons verwachten. Bij scheduling houdt het besturingssysteem hiermee rekening.

Real-time programma's: Real-time programmering legt aan de respons een tijdsbeperking op. Het wordt gebruikt wanneer een snelle respons essentieel is. Interactieve gebruikers geven de voorkeur aan een snelle respons, maar real-time gebruikers eisen dit zelfs. Voorbeelden van real-time verwerking: controlesysteem voor het luchtverkeer op een vliegveld, robots, ...



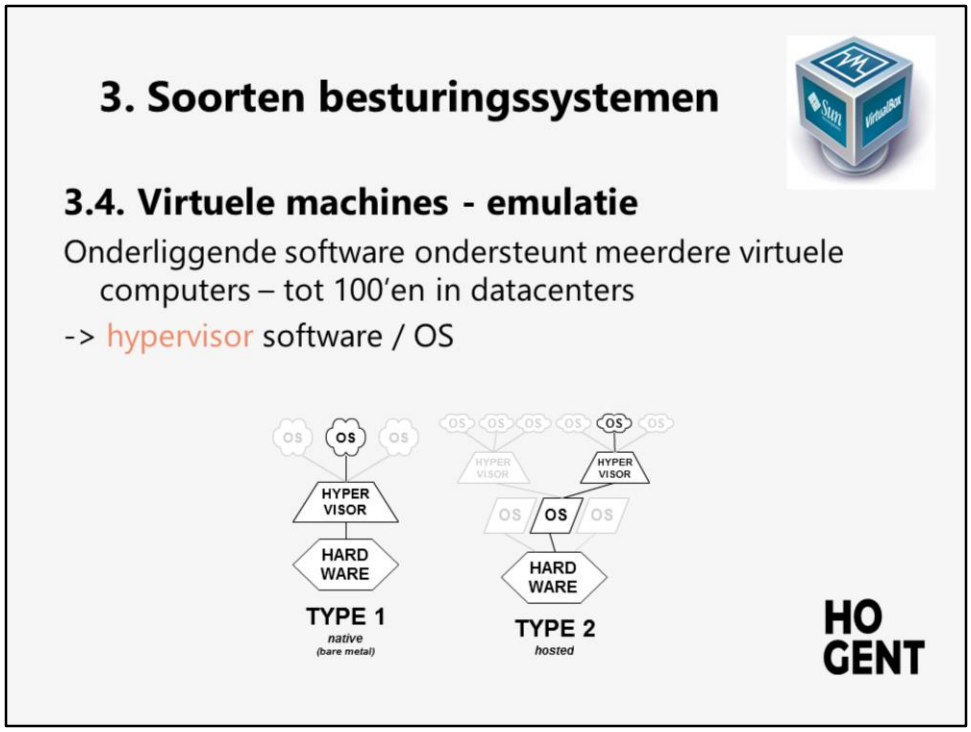
In het algemeen geven multiuser-systemen gebruikers de mogelijkheid, de verwerkingscapaciteit (resources) van een computer gemeenschappelijk te gebruiken. Omdat het besturingssysteem bepaalt welke resources er zijn, zien de gebruikers in het algemeen dezelfde resources. Er bestaat ook nog een ander type systeem, de virtual machine monitor.

(wikipedia)

Een **virtuele machine** is een computerprogramma dat een computer nabootst, waar andere programma's op kunnen worden uitgevoerd. Deze techniek wordt gebruikt om de overdraagbaarheid van programmatuur te verbeteren, en ook om het gelijktijdig gebruik van de computer door verschillende gebruikers robuuster te maken.

Virtuele machines zijn er in twee varianten: programmeertaal-specifiek of als emulator. **Programmeertaal-specifiek:** Een virtuele machine voor een programmeertaal biedt een *abstractielaag* voor de werkelijke computer: een verzameling basisfuncties waar programma's in de programmeertaal gebruik van moeten maken om de functies van de computer aan te spreken. Dit is precies wat een besturingssysteem doet, en een dergelijke machine fungeert dus als besturingssysteem voor in die taal geschreven programma's; de reden dat van een virtuele machine wordt gesproken is dat de virtuele machine zelf vaak een programma is dat wordt uitgevoerd op een bestaand besturingssysteem. De virtuele machine is de verbindende laag tussen de uitgevoerde code en de computerhardware (zoals de microprocessor) waarop het uiteindelijke programma wordt uitgevoerd, eventueel via een ander besturingssysteem. De op een virtuele machine uitgevoerde code spreekt niet direct de 'echte' hardware aan, maar gebruikt alleen de functies die worden aangeboden door de virtuele machine. Wanneer voor een bepaald platform (hardware en eventueel besturingssysteem) een virtuele machine gemaakt is, kan elk programma dat voor deze virtuele machine geschreven is, worden uitgevoerd. De implementatie van een virtuele machine kan in elke andere programmeertaal gebeuren. Op deze manier wordt platform-onafhankelijkheid bereikt: programma's kunnen worden uitgevoerd op elk systeem waarvoor de virtuele machine is geïmplementeerd. Overdraagbaarheid wordt een kwestie van het implementeren van een virtuele machine, in plaats van een probleem dat voor elk programma afzonderlijk moet worden opgelost. Een voorbeeld van een programmeertaal-specifieke virtuele machine is de **Java Virtual Machine** (JVM). Het resultaat van de compilatie van een Javaprogramma wordt bytecode genoemd. Een JVM kan dan deze bytecode-instructies één voor één interpreteren door ze te vertalen naar één of meerdere echte machine-instructies. Ook zijn er JVM's die de bytecode *just in time* (JIT) compileren naar de instructies van het platform. Dit levert uiteraard prestatiewinst op in vergelijking met interpretatie. Het idee van bytecode is niet recent. Sommige varianten van de programmeertaal Pascal hadden dit concept reeds; hier heette dit p-code of pseudocode. Ook veel Prolog implementaties zijn op een virtuele machine gebaseerd: de *Warren Abstract Machine*.

Emulatie: Een *virtuele machine* die de hardware van de computer emuleert, is niet gericht op een programmeertaal (zoals Java of Pascal), maar emuleert een fysieke computer (dat wil zeggen de processor en andere hardware), zodanig dat een bestaand besturingssysteem op deze emulatie kan draaien, alsof het een fysieke computer betreft. Met een dergelijke virtuele machine is het dus bijvoorbeeld mogelijk om Windows te emuleren in een GNU/Linux-omgeving, of omgekeerd.



Een **Hypervisor** of **Virtual Machine Monitor** is in de informatica een opstelling die ertoe dient om meerdere besturingssystemen tegelijkertijd op een hostcomputer te laten draaien. Met de term hypervisor wordt de eerdere term supervisor uitgebreid, die gewoonlijk toegepast werd op besturingssysteemkernels. Een hypervisor regelt een vorm van virtualisatie. Hypervisors zijn in twee soorten ingedeeld.

- Type1: 'Native' of 'Bare-Metal'
- Type2: 'Hosted'

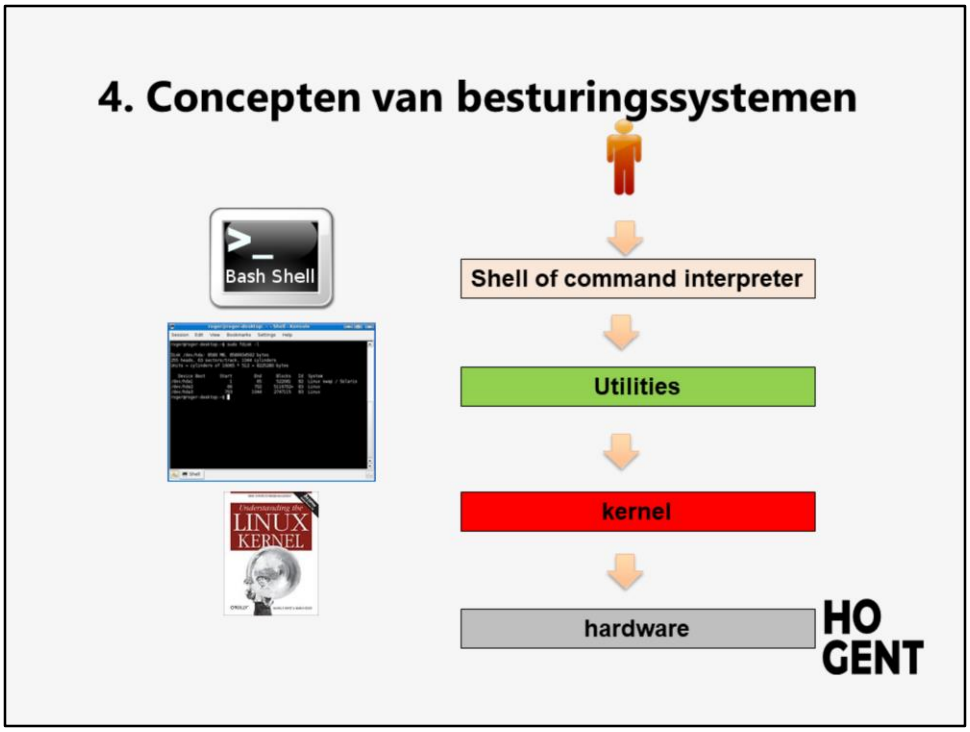
Type1

Een Type1 Hypervisor ligt direct op de hardware, dat wil zeggen dat er geen Besturingsysteem tussen ligt, hierdoor kunnen er meer resources aan de virtuele machines gegeven worden. een van de nadelen van een Type1 is dat je enige technische kennis moet hebben om er mee te kunnen werken. Type1 Hypervisors worden dan vooral ook gebruikt voor en door Servers.
Enkele voorbeelden van Type1 Hypervisors zijn: VMWare ESXi, Citrix Xen, KVM & Microsoft Hyper-V.

Type2

Een Type2 Hypervisor ligt in tegenstelling tot een Type1 niet direct tegen de hardware aan, dit wil zeggen dat er wel een Besturingsysteem onder ligt. Dit kunnen er verschillende zijn zoals: Microsoft Windows, Apple MacOS en Linux.
Voordelen die je hebt bij Type2 is dat hij aan de praat te krijgen is zonder al te veel technische kennis, meestal kunnen Type2 Hypervisors geïnstalleerd worden zoals een Programma. Een nadeel is dat Type2 niet zo krachtig en efficient zijn als Type1. Enkele voorbeelden van Type2 Hypervisors zijn: Oracle VirtualBox, VMware Workstation, Parallels Desktop.

Bron: https://nl.wikipedia.org/wiki/Hypervisor , geraadpleegd op 25/sept 2018

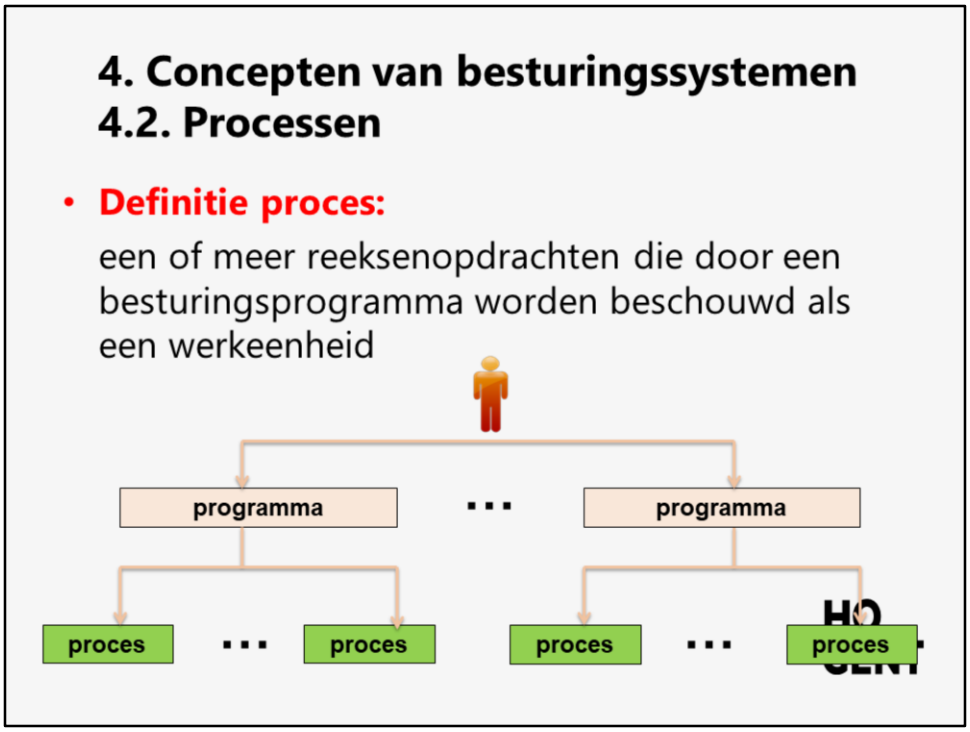


Veel besturingssystemen implementeren de interface tussen gebruiker en computer als een reeks stappen of lagen. In de toplaag zijn de functies vastgelegd en de onderste laag bevat de details van het laagste niveau om deze functies uit te voeren. De gebruiker communiceert met de bovenste laag. Deze bestaat uit routines van het besturingssysteem die zijn ontworpen om op opdrachten van een gebruiker te reageren. We noemen deze laag de shell of de command interpreter. Dit is het deel van het besturingssysteem waarmee de gebruiker het meest vertrouwd is.

In werkelijkheid is het echter niet de shell die de opdracht van de gebruiker uitvoert. De reden daarvoor is dat veel opdrachten eigenlijk erg ingewikkeld zijn. Zo is het mogelijk dat er voor een eenvoudige opdracht die het besturingssysteem vraagt veel moet gebeuren.

De laag utilities bevat vele routines die voor deze dingen zorg dragen.

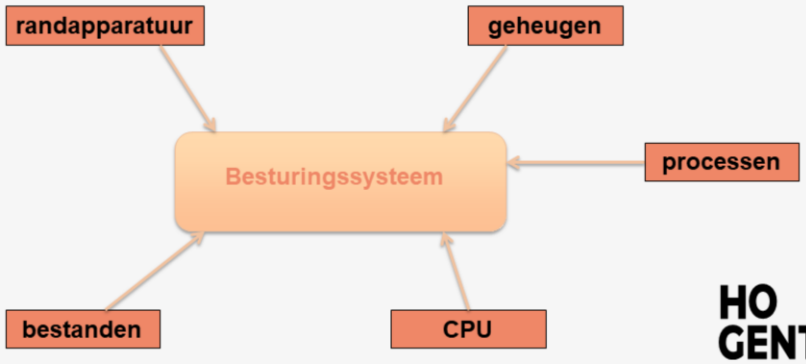
De laatste laag is de kernel of kern. Dit is het hart van het besturingssysteem. Deze laag bevat de routines die het vaakst worden gebruikt en waar het meest op aan komt. Wanneer een gebruiker een opdracht geeft, verzorgen de utilities het grootste deel van de controle en de voorbereiding die voor de uitvoering nodig zijn.



Wij moeten kijken hoe het besturingssysteem reageert op alle onafhankelijke activiteiten die kunnen voorkomen. We noemen een dergelijke activiteit een proces (task). Een proces is elke onafhankelijk uitgevoerde entiteit die meedingt naar het gebruik van resources.
Het aantal programma's per gebruiker en het aantal processen per programma varieert. Het besturingssysteem bekommert zich niet om de gebruiker en ook niet om het programma. Het systeem is in eerste instantie verantwoording schuldig aan de processen die worden uitgevoerd (runnen) en om resources strijden.

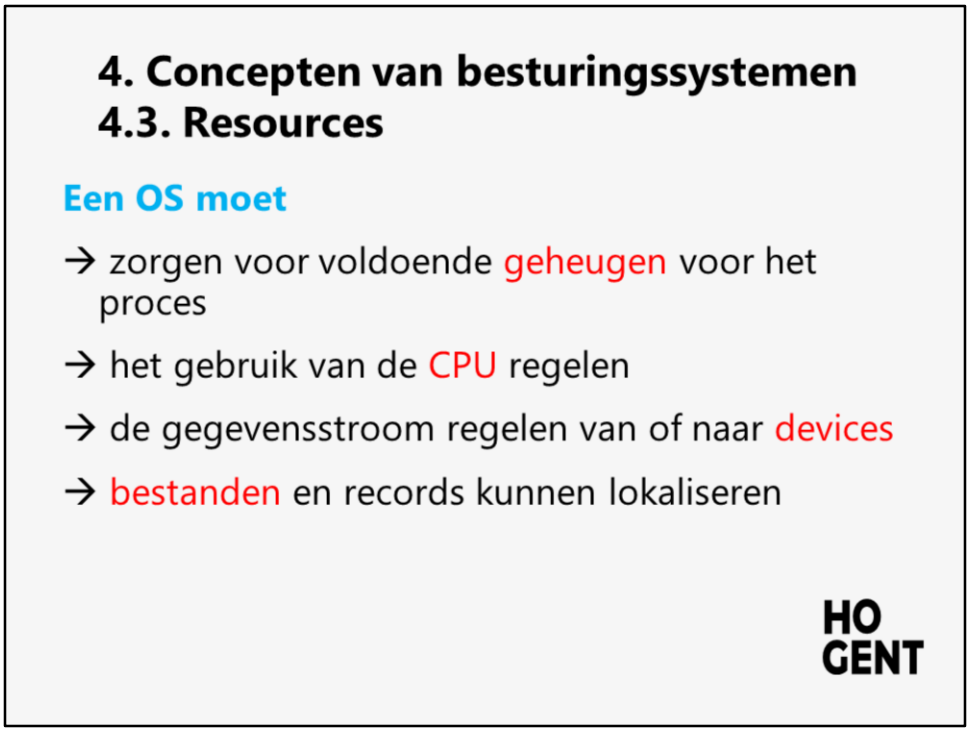
4. Concepten van besturingssystemen
4.3. Resources

→ in eerste instantie spreken processen recources (bronnen) aan



HO GENT

Het besturingssysteem moet open staan voor de behoeften van een proces. In de eerste plaats spreken processen resources aan.



Geheugen: Een proces heeft geheugen nodig waarin het zijn instructies en gegevens kan opslaan. Een besturingssysteem moet er daarom voor zorgen dat het proces voldoende geheugen krijgt. Geheugen is echter een eindige resource. Het besturingssysteem mag niet toelaten dat een proces zoveel geheugen heeft dat de andere processen niet kunnen "runnen". Bovendien stellen privacy en beveiliging de eis dat een proces niet in staat mag zijn zich eigenmachtig toegang tot het geheugen van een ander proces te verschaffen. Het besturingssysteem moet deze resource niet alleen toewijzen, maar ook de toegang regelen.
CPU: De CPU is ook een resource die elk proces nodig heeft om zijn instructies te kunnen uitvoeren. Aangezien er gewoonlijk meer processen dan CPU's zijn, moet het besturingssysteem het gebruik van de CPU regelen; dat moet echter wel eerlijk gebeuren. Als het goed is, krijgen belangrijke processen de CPU snel ter beschikking en mogen minder belangrijke processen de CPU niet gebruiken ten koste van andere.
Devices: Randapparatuur of devices zijn onder andere printers, tapedrives en disk-drives. Net als bij de CPU zijn er gewoonlijk meer gebruikers dan devices. Wat gebeurt er als verscheidene processen naar dezelfde printer of dezelfde drive willen schrijven? Het besturingssysteem moet uitzoeken wie tot wat toegang heeft. Het moet ook de gegevensstroom regelen wanneer de processen van devices lezen of naar devices schrijven.
Files: Het besturingssysteem wordt verondersteld snel een bepaalde file te kunnen lokaliseren. Ook wordt verwacht dat het snel een bepaald record in die file kan lokaliseren. Omdat devices vaak vele duizenden files bevatten en een file vele duizenden records kan bevatten, is dit een ingewikkelde taak.

4. Concepten van besturingssystemen
4.4. Concurrency

→ processen zijn meestal niet onafhankelijk, processen zijn concurrent

Bijvoorbeeld:

- 2 processen willen dezelfde printer gebruiken
- 2 processen willen dezelfde file lezen of schrijven
- 2 processen willen communiceren

=> conflicten!

→ OS regelt in welke volgorde processen afgehandeld worden = **synchronisatie**

HO GENT

4. Concepten van besturingssystemen
4.5. Ontwerp-criteria

- consistentie

- Flexibiliteit

- overdraagbaarheid

HO GENT

Het is duidelijk dat het beheer van resources en de uitvoer van de functies daarvan belangrijke criteria zijn bij het ontwerpen van besturingssystemen. Ook gebruiksgemak is een criterium.

Consistentie: Consistentie is een belangrijk ontwerp-criterium. Als het aantal processen, dat van de computer gebruik maakt, vrijwel constant blijft, hoort dat ook voor de respons te gelden.

Flexibiliteit: Een besturingssysteem hoort zo te zijn geschreven dat een nieuwe versie het draaien van oude applicaties niet onmogelijk maakt.

Bij een besturingssysteem moeten ook eenvoudig nieuwe randapparaten kunnen worden toegevoegd.

Overdraagbaarheid: Dit houdt in dat het besturingssysteem op verscheidene soorten computers werkt. Overdraagbaarheid geeft de gebruiker meer flexibiliteit.

4. Concepten van besturingssystemen
4.6. Compromissen

Vaak is het onmogelijk om aan alle criteria te voldoen en worden sommige opgeofferd ten gunste van het andere.

Al deze ontwerp-criteria zijn belangrijk; helaas is het meestal onmogelijk om aan alle te voldoen. Vaak moet het ene criterium worden opgeofferd ten gunste van het andere. De ontwerpers moeten de omgeving waarin het besturingssysteem werkt, grondig kennen. Op die manier kunnen ze bepalen welke criteria voor de gebruiker het belangrijkst zijn.